41장. 타이머

41.1 호출 스케줄링

1. **호출 스케줄링(scheduling a call)**

   : 함수를 명시적으로 호출하지 않고 일정 시간이 경과된 이후에 호출되도록 함수 호출을 예약

2. 자바스크립트의 타이머를 생성/제거할 수 있는 타이머 함수

   생성 (생성한 타이머가 만료되면 콜백 함수 호출)

   - setTimeout (타이머가 만료되면 한 번 호출)
   - setInterval (타이머가 만료될 때마다 반복 호출)

   삭제

   - clearTimeout
   - clearInterval

- 타이머 함수는 빌트인 함수가 아닌, **호스트 객체**

3. 자바스크립트 엔진은 단 하나의 실행 컨텍스트 스택을 갖기 때문에 두 가지 이상 동시 실행 X

   ➡️ 즉, JS 엔진은 **싱글 스레드(single thread)**로 동작

   이런 이유로 타이머 함수 setTimeout, setInterval은 **비동기(asynchronous) 처리 방식**으로 동작

41.2 타이머 함수

41.2.1 setTimeout / clearTimeout

1. 두 번째 인수로 전달받은 시간으로 단 한 번 동작하는 타이머를 생성.

   이후 타이머가 만료되면, 첫 번째 인수로 전달받은 콜백 함수 호출

```
// 1초(1000ms) 후 타이머가 만료되면 콜백 함수 호출
setTimeout(() => console.log('Hi!'), 1000);

// 1초(1000ms) 후 타이머가 만료되면 콜백 함수 호출
// 이때 콜백 함수에 'Lee'가 인수로 전달
setTimeout(name => console.log(`Hi! ${name}.`), 1000, 'Lee');
```

```
// 두 번째 인수(delay)를 생략하면 기본값 0이 지정
setTimeout(() => console.log('Hello!'));
```

2. setTimeout 함수가 반환한 타이머 id를 clearTimeout 함수의 인수로 전달하여 타이머를 취소

   즉, clearTimeout 함수는 호출 스케줄링을 취소.

41.2.2 setInterval/ clearInterval

1. 두 번째 인수로 전달받은 시간으로 반복 동작하는 타이머를 생성.

   이후 타이머가 만료되면, 첫 번째 인수로 전달받은 콜백 함수 반복 호출

```
let count = 1;

// 1초(1000ms) 후 타이머가 만료될 때마다 콜백 함수 호출
// setInterval 함수는 생성된 타이머를 식별할 수 있는 고유한 타이머 id 반환
const timeoutId = setInterval(() => {
    console.log(count);  // 1 2 3 4 5

    // count가 5되면 타이머를 취소
    if ( count++ === 5 ) clearInterval(timeoutId);
}, 1000);
```

2. setInterval 함수가 반환한 타이머 id를 clearInterval 함수의 인수로 전달하여 타이머를 취소

   즉, clearInterval 함수는 호출 스케줄링을 취소.

41.3 디바운스와 스로틀

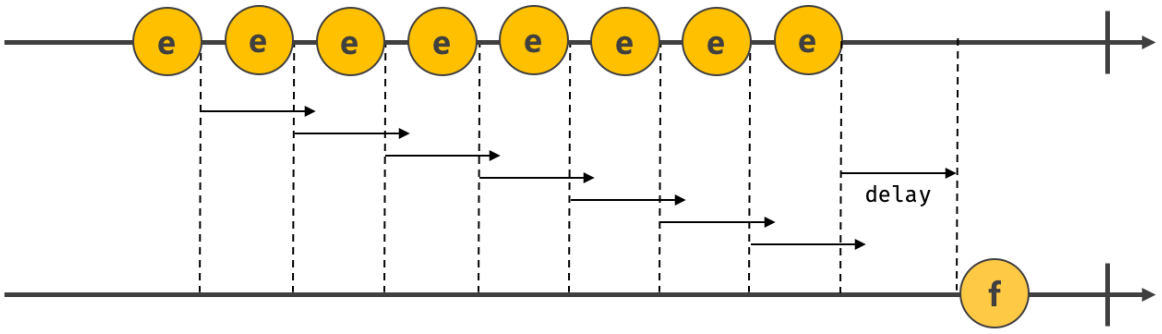
1. 디바운스 & 스로틀

   : 짧은 시간 간격으로 연속해서 발생하는 이벤트를 그룹화해서 과도한 이벤트 핸들러의 호출을 방지하는 프로그래밍 기법

41.3.1 디바운스

1. 디바운스(debounce)

: 짧은 시간 간격으로 이벤트가 연속 발생하면, 이벤트 핸들러를 호출하지 않다가 일정 시간이 경과한 후에 이벤트 핸들러가 한 번만 호출되도록 함.



2. 유용하게 사용되는 곳
   - resize 이벤트 처리
   - input 요소에 입력된 값으로 ajax 요청하는 입력 필드 자동완성 UI 구현
   - 버튼 중복 클릭 방지 처리

3. 실무에서는 Underscore의 debounce 함수, Lodash의 debounce 함수 사용을 권장.

41.3.2 스로틀

1. 스로틀(throttle)

   : 짧은 시간 간격으로 이벤트가 연속 발생하더라도, 일정 시간 간격으로 이벤트 핸들러가 최대 한 번만 호출되도록 함.

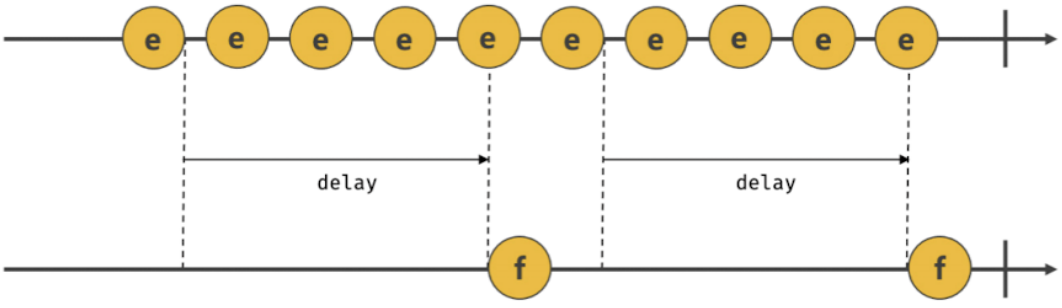


그림 41-3 스로틀

2. 유용하게 사용되는 곳
   - scroll 이벤트 처리

- 무한 스크롤 UI 구현

3. 실무에서는 Underscore의 throttle 함수, Lodash의 throttle 함수 사용을 권장.